Executable Disease Networks using CaSQ

Anna Niarakis and **Sylvain Soliman**

Inria

Computational Systems Biology for Complex Human Disease:
From static to dynamic representations of disease mechanisms

Sylvain Soliman

Computer Science Researcher

Inria Saclay Île de France, Lifeware team

Biography

Since 2002 I am a permanent researcher (CR) in the Lifeware (formerly Contraintes) group of Inria Saclay-Île-de-France (formerly Paris-Rocquencourt).

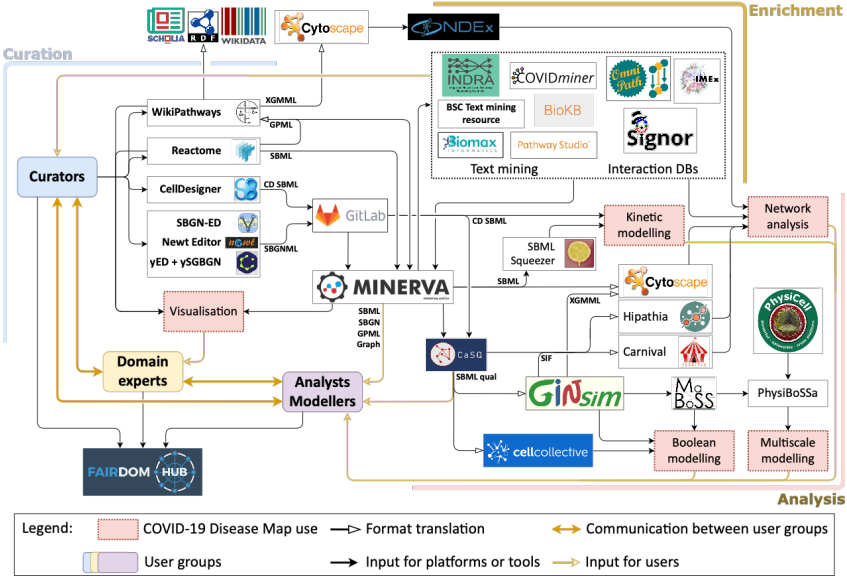
My research interests focus around Computational Biology and Theoretical Computer Science. In this context I'm one of the main developers and maintainers of the BIOCHAM platform. This is where most of the techniques I develop, using Constraint Programming, Model-Checking, and other formal methods get implemented.

On top of my BIOCHAM developments, you can find below other side projects concerning Vim, Prolog, etc. I used to maintain separately Nicotine, a constraint-based software for Petri-net invariant computation and Tropical equilibration, but it has now been completely merged into BIOCHAM.

I've been teaching since 2005 in the MPRI and just started in 2018 teaching in the AI&AVC Master. You can find below more information.

I'm a member of Inria Saclay's Scientific Commission, in 2016 I was also a member of the jury for AAP Digiteo/Digicosme Ph.D. grants. When I was in Rocquencourt I was president of the Doctoral Committee for quite some time, and of the Technological Development Commission for a few years.

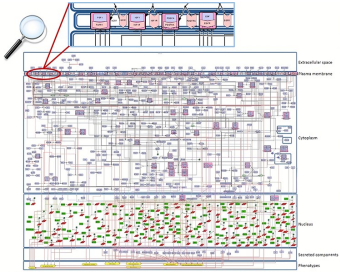
C19DM Ecosystem



Using CaSQ to bridge a gap...

People curate data/literature to build large, rich and complex knowledge maps

How do you compare them with experiments?

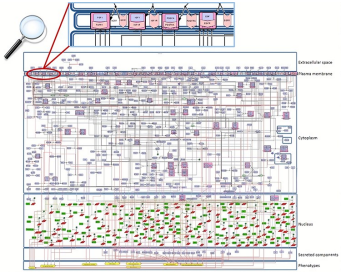


Map

Using CaSQ to bridge a gap...

People curate data/literature to build large, rich and complex knowledge maps

How do you compare them with experiments?

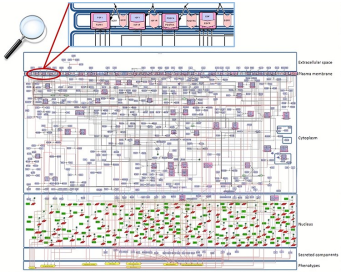


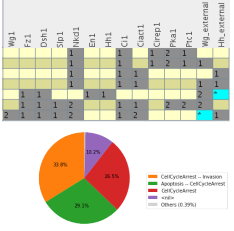
Map

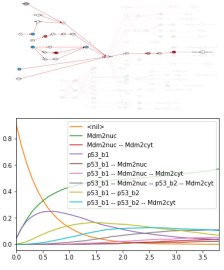
Using CaSQ to bridge a gap...

People curate data/literature to build large, rich and complex knowledge maps

How do you compare them with experiments?







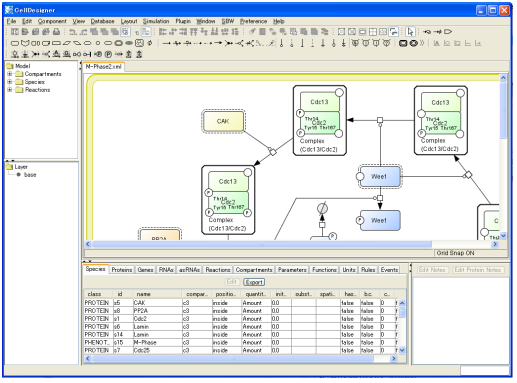
Map ⟹ Model

Molecular Interaction Maps (Monday sessions)

Built from scratch

Starting from an available map (e.g., Rheumatoïd Arthritis, Covid-19 DM, Atlas of Cancer Signalling Network)

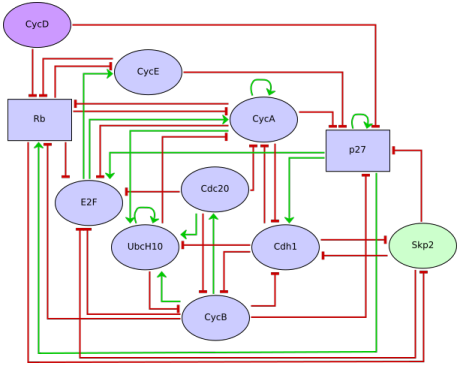
Reactome, KEGG → MINERVA
Translated into SBGN-PD



Described as *reusable* and *comprehensive* Process Description diagrams: detailed, mechanistic

Encoded in SBML using CellDesigner: rich annotations

Logical Models *à la* René Thomas



Qualitative abstraction:

- Discrete (*Boolean*) state
- **Dynamics ⇐ logical formulae**
- *Logical* time
- Phenotype ≡ *steady state*

More details in today's other sessions...

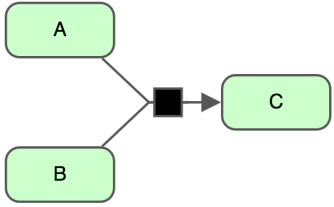
What CaSQ does

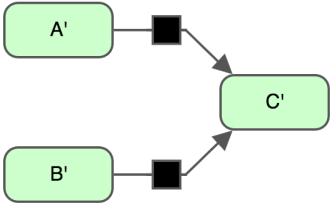
- Handle even very large detailed mechanistic maps
- Keep and use all information in the annotations
- Build a fully executable/dynamic logical model

What CaSQ does not do

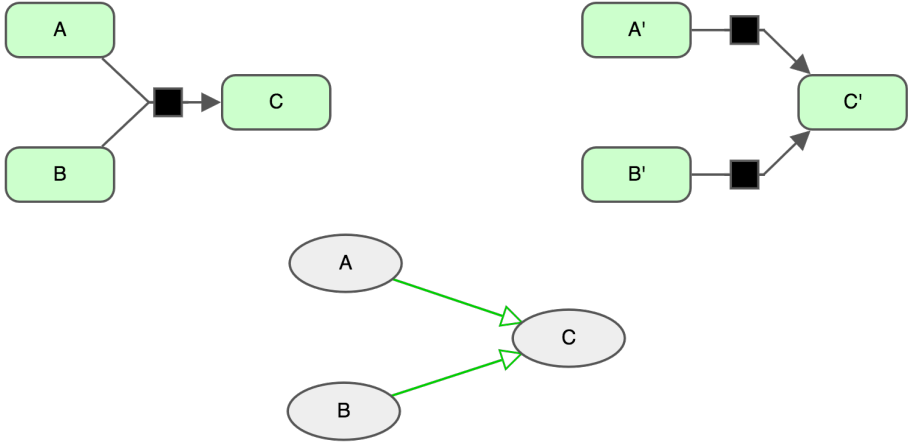
- extract a *relevant* submap
  (CaSQ can extract up/down-stream however)
- only convert PD to AF
  (he who can do the most can do the least,
  CaSQ saves the AF as .sif file for Cytoscape/Hipathia)
- build a numerical (ODE-based) dynamic model
- build *the best* logical model
- handle the analysis of the model

The AF graph loses information



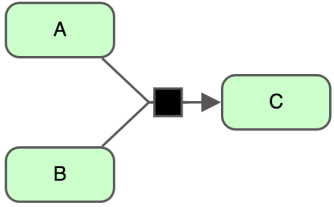


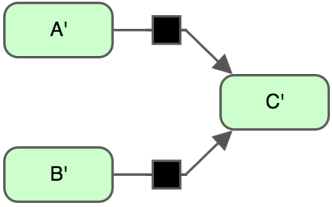
The AF graph loses information

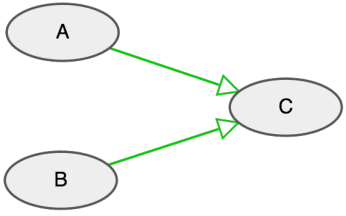


Same Activity Flow,

The AF graph loses information



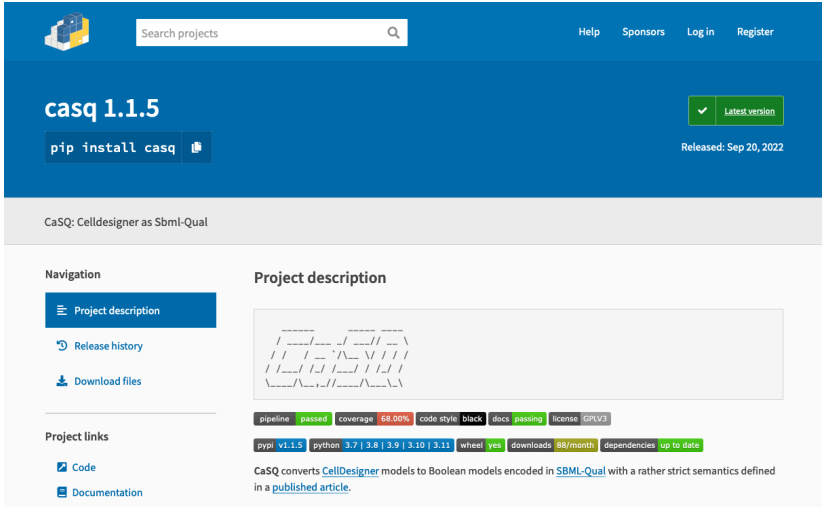




Same Activity Flow, but different Logical Models ($\wedge$ vs. $\vee$)

Installing CaSQ (from PyPI)

```
python3 -m pip install casq
```



Usage (CLI or directly from Python, e.g., in CoLoMoTo)

```
casq -h
```

```
usage: casq [-h] [-v] [-D] [-c] [-s] [-r S] [-u [UPSTREAM ...]] [-d [DOWNSTREAM ...]] [-b] [-g
    GRANULARITY] [-i INPUT] [-C]
           [infile] [outfile]

Convert CellDesigner models to SBML-qual with a rather strict semantics. Copyright (C) 2019-2022
    Sylvain.Soliman@inria.fr GPLv3

positional arguments:
  infile                CellDesigner File
  outfile               SBML-Qual/BMA json File

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit
  -D, --debug           Display a lot of debug information
  -c, --csv             Store the species information in a separate CSV file
  -s, --sif             Store the influence information in a separate SIF file
  -r S, --remove S      Delete connected components in the resulting model if their size is
      smaller than S. A negative S leads to
                        keep only the biggest(s) connected component(s)
  -u [UPSTREAM ...], --upstream [UPSTREAM ...]
                        Only species upstream of this/these species will be kept
  -d [DOWNSTREAM ...], --downstream [DOWNSTREAM ...]
                        Only species downstream of this/these species will be kept
```

A 3-step process

- map simplification (4 rules)
- logical model (logical rules and graph) computation
- model simplification

(S. S. Aghamiri et al. Bioinformatics 36 (16) Aug. 2020)

Step 2 — General Logical Rule

Using *inhibitor* and other modifier **annotations**:

"A target is on when any of the reactions producing it is on

a reaction is on if all reactants are on, all inhibitors are off and one of the catalysts is on if there are any."

$$t = \bigvee_{R,C,I \to t\ldots} \left(\bigwedge_{r \in R} r \bigwedge_{i \in I} \neg i \bigvee_{c \in C \text{ or } \{\top\} \text{ if } C=\varnothing} c \right)$$

R POSITIVE *t*
C POSITIVE *t*
I NEGATIVE *t*

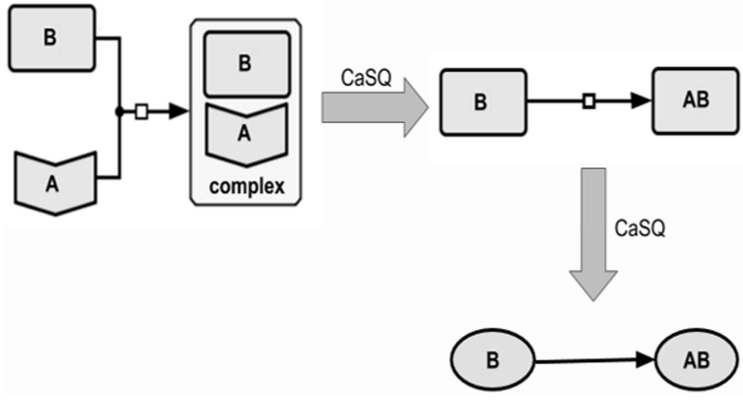
Step 3 — Cleanup

Maps are **not connected**
Keep only some CCs (according to their size)

Names are **ambiguous**
Rename species if necessary (e.g., type, modifications and compartment disappear in the model)

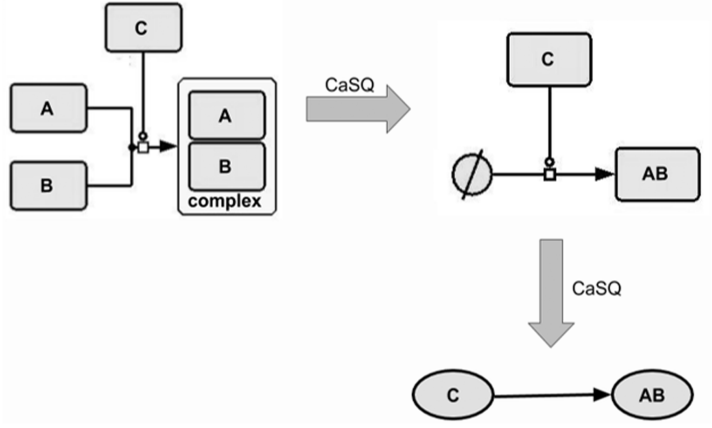
Generate auxiliary files if requested (SIF, CSV, BMA...)

Step 1, Rule 1 — Receptor Deletion



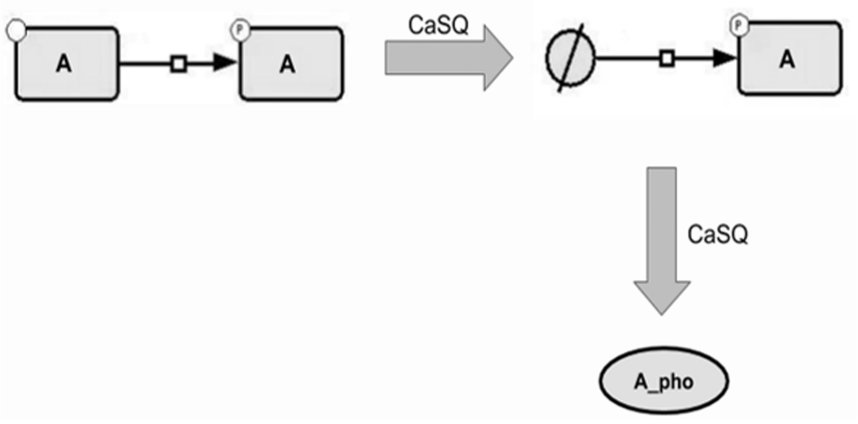
A and B do not take part in any other reaction. A annotated as **receptor**, reaction as *heterodimer association*

Step 1, Rule 2 – Complex Merge



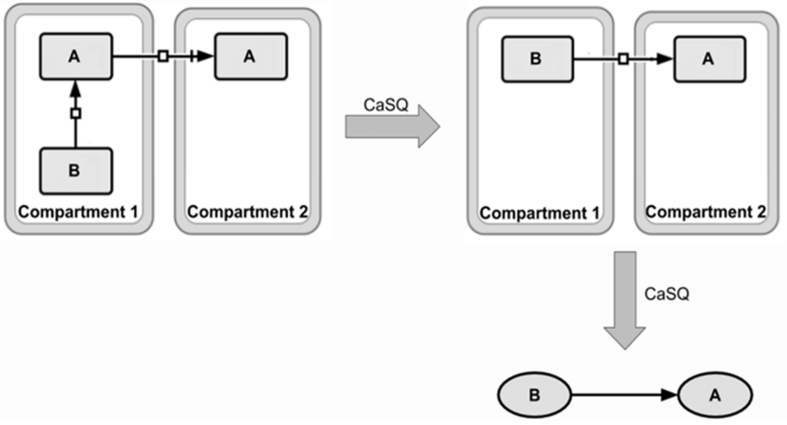
A and B do not take any part in any other reaction, reaction annotated as **heterodimer association**

Step 1, Rule 3 — Inactive Species Removal



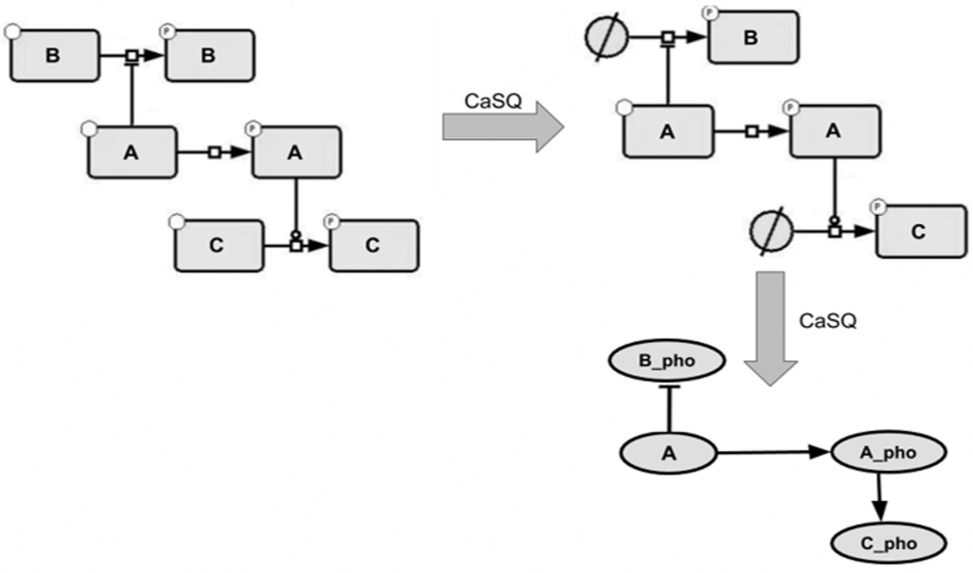
A only appears in this reaction with a single product, both reactant and product have the **same name**

Step 1, Rule 4 — Transport Simplification

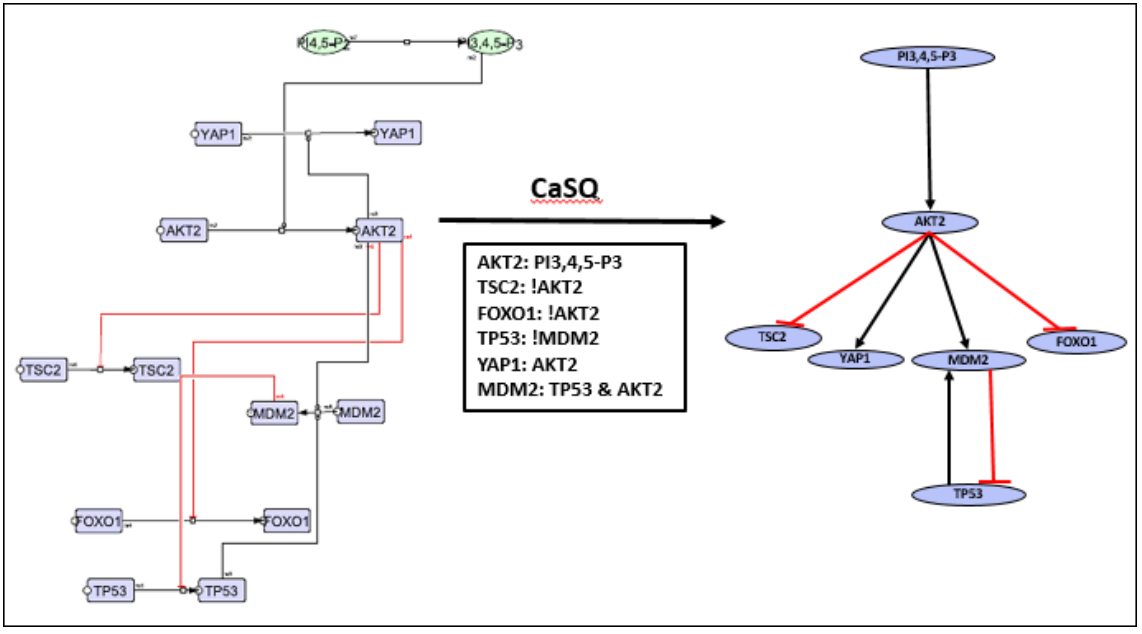


A (reactant) does not take any other *active* part, reaction annotated as **transport**, *same name* in both compartments

Example (practical reduction is about 30%)



RA map excerpt (50% reduction)



Conserving *information*

All vertices of the model correspond to one (species) vertex of the original map

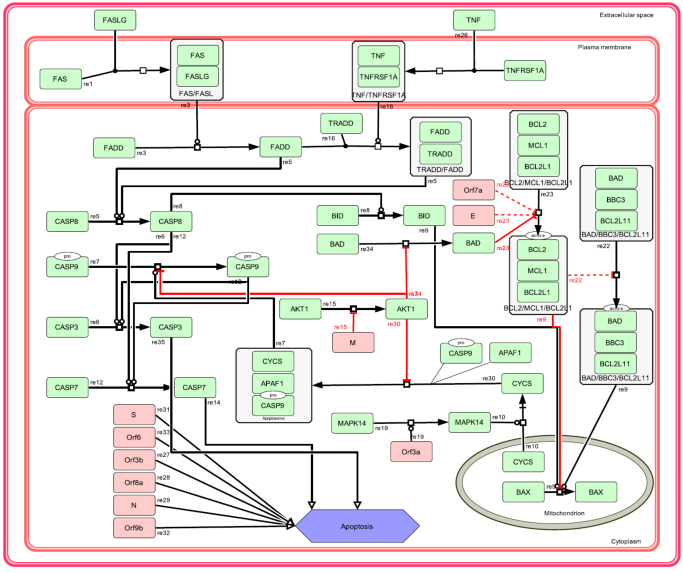
Let us **keep its original layout** for readability!

Annotations about the nature of a species (e.g., receptor) are used in the transformation

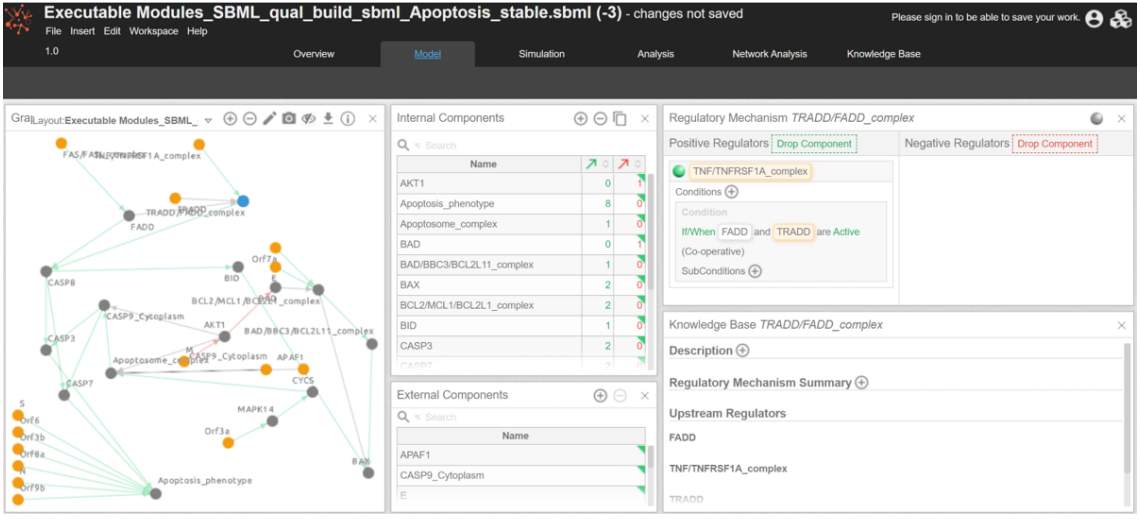
But there is much more, e.g., bibliographic information
MIRIAM: Minimal Information Required In the Annotation of Models

Propagate that of vertices that are kept. Merge that of removed vertices into a nearby meaningful vertex

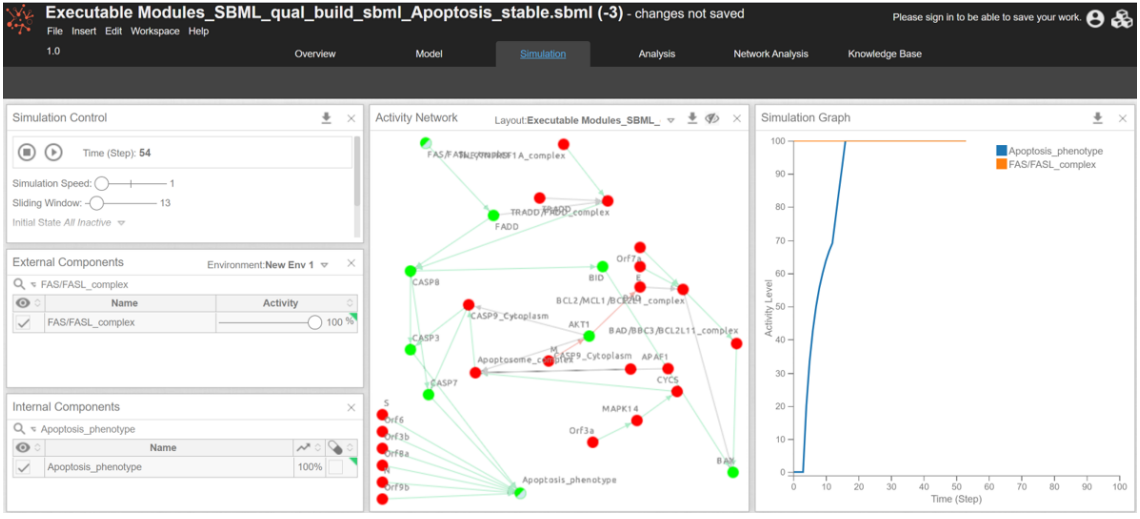
Apoptosis module (C19DM)



Apoptosis model in TCC



Apoptosis model in TCC



SBML in, SBML out

Standards are crucial for interoperability

Rich annotations “in”

Rich annotations “out”

Conclusion

CaSQ automatically produces a
dynamic executable model from a static map

It strongly relies on annotations and topology to infer logical rules

It embraces SBML standards in and out for interoperability

It conserves rich annotations and layout, even when reducing the model, for readability and reusability

This does not lead to a *perfect* model, but is scalable, fast and consistent for a first step

TL;DL

”The map is not the territory.”

— Alfred Korzybski

TL;DL

”The map is not the territory.”
— Alfred Korzybski

”All models are wrong, but some are useful.” — George Box

C19DM Ecosystem